Under consideration for publication in Math. Struct. in Comp. Science

Imaginary groups: lazy monoids and reversible computation

Murdoch J. Gabbay[1] and Peter H. Kropholler[2]

[1] *School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, United Kingdom*
`http://www.gabbay.org.uk`

[2] *Mathematical Sciences, Southampton University, Highfield, Southampton SO17 1BJ, United Kingdom*
`p.h.kropholler@southampton.ac.uk`[†]

Received 30 September 2011; Revised 21 April 2012

By constructions in monoid and group theory we exhibit an adjunction between the category of partially ordered monoids and *lazy* monoid homomorphisms, and the category of partially ordered groups and group homomorphisms, such that the unit of the adjunction is injective. We also prove a similar result for sets acted on by monoids and groups.
We introduce the new notion of *lazy homomorphism* for a function f between partially-ordered monoids such that $f(m \circ m') \leq f(m) \circ f(m')$.
Every monoid can be endowed with the discrete partial ordering ($m \leq m'$ if and only if $m = m'$) so our constructions provide a way of embedding monoids into groups. A simple counterexample (the two-element monoid with a non-trivial idempotent) and some calculations show that one can never hope for such an embedding to be a monoid homomorphism, so the price paid for injecting a monoid into a group is that we must weaken the notion of homomorphism to this new notion of lazy homomorphism.
The computational significance of this is that a monoid is an abstract model of computation—or at least of 'operations'—and similarly a group models *reversible* computations/operations. By this reading, the adjunction with its injective unit gives a systematic high-level way of faithfully translating an irreversible system to a 'lazy' reversible one.
Informally, but perhaps informatively, we can describe this work as follows: we give an abstract analysis of how to sensibly add 'undo' (in the sense of 'control-Z').

[†] Many thanks to Phil Scott and to an anonymous referee, without whom the paper would not have attained its current form. The first author acknowledges the support of the Leverhulme trust and of grant RYC-2006-002131 at the Polytechnic University of Madrid; the second author acknowledges the support of Cornell University, New York and of the Institut Mittag-Leffler, Sweden.

Contents

1 Introduction 2
1.1 Monoids, groups, computation, and inverses 2
1.2 Monoids, lazy monoids, and our solution 3
1.3 Map of the paper 4
2 Basic definitions 5
2.1 Partially ordered groups and monoids, and lazy homomorphisms 5
2.2 Why lazy? 6
3 The imaginary group 7
3.1 The free group 8
3.2 Positive cones 9
3.3 The partial order 10
3.4 Technical lemmas 11
3.5 The preorder is a partial order and the natural map is injective 13
3.6 The functor from PoMonZ to PoGrp 14
3.7 The forgetful functor from PoGrp to PoMonZ 15
4 Commutative and nonassociative generalisation 16
4.1 Commutative and other generalisations 17
4.2 Generalisation to categories and groupoids 17
5 Sets with an action 18
5.1 Construction of the sets with an action 19
5.2 The natural adjunction 20
5.3 More detailed analysis for the partial order 22
6 An example: substitutions 23
6.1 Substitutions just on variables 23
6.2 Substitutions on terms 23
7 Conclusions 24

1. Introduction

1.1. *Monoids, groups, computation, and inverses*

Suppose we have a computer. We run programs, and this changes the state of the computer.

This can be thought of as a set X (representing *states*) and some functions $f : X \to X$ (representing *operations*) closed under function composition. The functions in $X \to X$ form a *monoid*, which is like a group but does not have inverses. Monoids provide an abstract model of state and computation.

Now suppose we want to undo the effects of an operation. But since $f : X \to X$ need not be injective, given some $x \in X$ there may be several y such that $f(y) = x$. Monoids do not have inverses.

Yet 'undo' or 'rollback' operations are very common. This may be familiar to the

reader from e.g. word processing or version control (subversion was used in the production of this paper). A notion of 'state, with rollback' called *transactional memory* is useful, especially in concurrency where updates from one process may need to be undone in the light of results later arrived at by a concurrent process [Herlihy and Moss, 1993]. In proof-search the difference between proof-rules which are reversible, such as left conjunction introduction $(\wedge\mathbf{L})$ below, also called *asynchronous*, and those that are not reversible, such right disjunction introduction $(\vee\mathbf{R})$ below, also called *synchronous*, has mathematical significance [Liang and Miller, 2009].

$$\textit{Asynchronous}\quad \frac{\Gamma, A\wedge B\vdash C}{\Gamma, A, B\vdash C}\,(\wedge\mathbf{L}) \qquad \textit{Synchronous}\quad \frac{\Gamma\vdash A}{\Gamma\vdash A\vee B}\,(\vee\mathbf{R})$$

Note that $(\wedge\mathbf{L})$ just rearranges connectives (by turning a '$\wedge$' into a ',' if we read the rule bottom-up), but $(\vee\mathbf{R})$ eliminates B so is (from the point of view of proof-search) an irreversible step.

Further, the field of reversible computation, which has physical importance for heat-dissipation and the design of biological and quantum computation, is devoted to the study of hand-tailored reversible computational mechanisms. References and further discussion are in the Conclusions.

Now let us return to our monoid $\mathcal{M}$. This is given to us and it was *not* designed from the start to be reversible. How can we 'add an undo function' to $\mathcal{M}$, in some abstract sense? In view of the discussion above we can mathematically distil this to the following formal and rigorously answerable question:

Given a monoid $\mathcal{M}$, does there exist a group $\mathcal{M}^{\pm}$ and an **injective** *monoid map $\mathcal{M}\to\mathcal{M}^{\pm}$?*

We ask for the map to be injective so that distinct operations in the monoid correspond to distinct *reversible* operations in the group. That is, $\mathcal{M}^{\pm}$ needs to faithfully represent the original states and operations. Otherwise, we could just map all of $\mathcal{M}$ to the trivial group with one element.

Given the injection above, we could have our cake and eat it: given any notion of computation, represent it as a monoid $\mathcal{M}$, calculate $\mathcal{M}^{\pm}$, compute in that group (with an undo function) and then, if we like the result, we can return our result and if we do not, we can always undo.

1.2. *Monoids, lazy monoids, and our solution*

Unfortunately what we have asked for is impossible.

This can easily be proved with a small counterexample:

Example 1.1. Consider the two-element monoid $\{0,1\}$ with

$$1\circ x = x \quad\text{for}\quad x\in\{0,1\} \quad\text{and}\quad 0\circ x = 0 \quad\text{for}\quad x\in\{0,1\}.$$

If we try to adjoin an element 0^{-1} then $0 = 0\circ 0$ and so (applying 0^{-1} to both sides) $0 = 1$. We lose injectivity; if $\mathcal{M} = \{0,1\}$ then $\mathcal{M}^{\pm}$ is the trivial group in which computation is reversible because everything is equal to the identity operation which does nothing at all.

In this paper we will indeed exhibit a construction which assigns to a monoid $\mathcal{M}$ a *partially ordered* group $\mathcal{M}^{\pm}$, and we will exhibit a *lazy* monoid homomorphism from $\mathcal{M}$ to $\mathcal{M}^{\pm}$. By moving to partially ordered structures and lazy homomorphisms, the contradiction noted in the previous paragraph is avoided.

Thus it is possible to add 'undo' to a monoid after all—we just have to do it in the right way. How this should work, is not immediately obvious.

Our solution supports an attractive computational interpretation: think of the monoid $\mathcal{M}$ as a set of operations, and monoid composition as their composition, and both of these may be irreversible. Then the 'lazy' group $\mathcal{M}^{\pm}$ which we build, is operations which have been assembled but not yet carried out; they are 'suspended'. Each element in the group is a plan of action; we can compose plans, modify them, erase elements from them—and this is reversible because nothing has been carried out. The partial order represents the operation of actually carrying out the plan; moving down along the partial order represents refining results as the plan is executed, which may lose information and is irreversible.

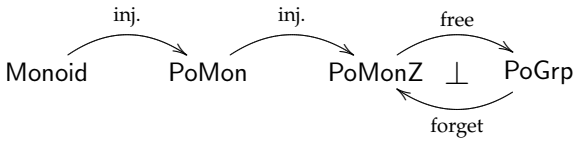
Our main result can be stated as follows:

There is an adjunction between the categories of partially-ordered monoids and lazy homomorphisms, and of partially-ordered groups and group homomorphisms, and the unit of the adjunction is injective.

This is Theorems 3.22, 3.24, 3.25, and 3.30.

1.3. *Map of the paper*

Here is an eagle's eye view of how our construction works for monoids:



Monoid is the category of monoids and monoid homomorphisms. PoMon is the category of partially-ordered monoids and monotone monoid homomorphisms. The injection from Monoid to PoMon is just obtained by giving a monoid the trivial partial order relating x to itself.

PoMonZ is the category of partially-ordered monoids and lazy monotone monoid homomorphisms, where 'lazy' is defined in Definition 2.3. Every monotone monoid homomorphism is a lazy monotone monoid homomorphism so we obtain a trivial injection from PoMon to PoMonZ.

PoGrp is the category of partially-ordered groups and monotone group homomorphisms. There is an easy forgetful functor from PoGrp to PoMonZ and a *free* functor that is left adjoint to the forgetful functor mapping a monoid $\mathcal{M}$ to its free group with added 'imaginary' inverse $\mathcal{M}^{\pm}$. Constructing the underlying set of $\mathcal{M}^{\pm}$ easy, as is defining its group structure and giving it a natural preorder (Definitions 3.3 and 3.6). What is not easy is proving that the preorder is a partial order, and showing that the unit of the adjunction is a monomorphism. This is Theorems 3.22 and 3.24, see also Theorem 3.25.

Later on in Section 4 we discuss how this can be generalised in various ways, and in Section 5 we extend the construction to sets with a lazy monoid or group action. The generalisations are essentially routine, and extending to sets with an action is not entirely trivial, but no harder.

2. Basic definitions

2.1. *Partially ordered groups and monoids, and lazy homomorphisms*

Definition 2.1. Consider the following axioms:

$$\begin{array}{ll} (\textbf{assoc}) & x \circ (x' \circ x'') = (x \circ x') \circ x'' \\ (\textbf{id}) & \mathrm{id} \circ x = x \circ \mathrm{id} = x \\ (\textbf{inv}) & x \circ x^{-1} = x^{-1} \circ x = \mathrm{id} \end{array}$$

— A **group** $\mathcal{G}$ is a tuple $(|\mathcal{G}|, \circ, \mathrm{id}, ^{-1})$ of a **carrier set** $|\mathcal{G}|$, a **composition** $\circ$ mapping $|\mathcal{G}| \times |\mathcal{G}|$ to $|\mathcal{G}|$, an element $\mathrm{id} \in |\mathcal{G}|$, and an **inverse map** $^{-1}$ mapping $|\mathcal{G}|$ to $|\mathcal{G}|$, satisfying (**assoc**), (**id**), and (**inv**).
— A **monoid** $\mathcal{M}$ is a tuple $(|\mathcal{M}|, \circ, \mathrm{id})$ as above, satisfying (**assoc**) and (**id**).

Definition 2.2. A **(compatible) order** on a monoid $\mathcal{M}$ is a partial order $\leq \subseteq |\mathcal{M}| \times |\mathcal{M}|$ such that $x \leq y$ implies $x \circ z \leq y \circ z$ and $z \circ x \leq z \circ y$.

— A **partially ordered monoid** is a pair $(\mathcal{M}, \leq)$ of a monoid $\mathcal{M}$ and a compatible order on $\mathcal{M}$.
— A **partially ordered group** is a pair $(\mathcal{G}, \leq)$ of a group $\mathcal{G}$ and a compatible order on $\mathcal{G}$, considered as a monoid.

Definition 2.3. Call a function $f \in |\mathcal{M}| \to |\mathcal{M}'|$ between monoids $\mathcal{M}$ and $\mathcal{M}'$ a **homomorphism** when it satisfies the two axioms

$$\mathrm{id}_{\mathcal{M}'} = f(\mathrm{id}_{\mathcal{M}}) \qquad \text{and} \qquad f(x_1 \circ_{\mathcal{M}} x_2) = f(x_1) \circ_{\mathcal{M}'} f(x_2).$$

If $\mathcal{M}'$ is a group, the first equation can be deduced from the second, as can the identity $f(x^{-1}) = f(x)^{-1}$; but in general both axioms are required.

Suppose now that $\mathcal{M}$ and $\mathcal{M}'$ are partially ordered monoids. Call a function $f \in |\mathcal{M}| \to |\mathcal{M}'|$ a **lazy homomorphism** when

$$\mathrm{id}_{\mathcal{M}'} \leq_{\mathcal{M}'} f(\mathrm{id}_{\mathcal{M}}) \qquad \text{and} \qquad f(x_1 \circ_{\mathcal{M}} x_2) \leq_{\mathcal{M}'} f(x_1) \circ_{\mathcal{M}'} f(x_2).$$

Call $f \in |\mathcal{M}| \to |\mathcal{M}'|$ **monotone** when $x \leq_{\mathcal{M}} y$ implies $f(x) \leq_{\mathcal{M}'} f(y)$.

We discuss this definition in Subsection 2.2.

Definition 2.4.

— Write PoMonZ for the category of partially ordered monoids and monotone lazy homomorphisms between them *(Partially Ordered MONoids with laZy homomorphisms)*.

— Write PoGrp for the category of partially ordered groups and monotone lazy group homomorphisms between them *(Partially Ordered GRouPs and homomorphisms)*.

Note in Definition 2.4 that we take lazy homomorphisms in PoMonZ but 'real' homomorphisms in PoGrp. We could certainly define PoGrpZ, but it turns out that we will never use it; see Example 3.28 for an example illustrating why this is the case.

Remark 2.5. For the reader's convenience we explicitly unpack what it is to be an arrow in PoMonZ. An arrow $f : (\mathcal{M}, \leq_{\mathcal{M}}) \to (\mathcal{M}', \leq_{\mathcal{M}'})$ is a function f from $|\mathcal{M}|$ to $|\mathcal{M}'|$ such that:

1 $\mathrm{id}_{\mathcal{M}'} \leq_{\mathcal{M}'} f(\mathrm{id}_{\mathcal{M}})$ (*laziness*).
2 $f(x \circ_{\mathcal{M}} y) \leq_{\mathcal{M}'} f(x) \circ_{\mathcal{M}'} f(y)$ (*laziness*).
3 $x \leq_{\mathcal{M}} y$ implies $f(x) \leq_{\mathcal{M}'} f(y)$ (*monotonicity*).

2.2. *Why lazy?*

There is a notion of *lax functor* between 2-categories (see for instance [Leinster, 2004, Definition 1.5.8]). Specifically for us here,[†] this suggests the following definition:

Definition 2.6. Suppose now that $\mathcal{M}$ and $\mathcal{M}'$ are partially ordered monoids. Call a function $f \in |\mathcal{M}| \to |\mathcal{M}'|$ a **lax homomorphism** when

$$f(\mathrm{id}_{\mathcal{M}}) \leq_{\mathcal{M}'} \mathrm{id}_{\mathcal{M}'} \qquad \text{and} \qquad f(x_1 \circ_{\mathcal{M}} x_2) \leq_{\mathcal{M}'} f(x_1) \circ_{\mathcal{M}'} f(x_2).$$

Call a function $f \in |\mathcal{M}| \to |\mathcal{M}'|$ an **oplax** or **colax homomorphism** when

$$\mathrm{id}_{\mathcal{M}'} \leq_{\mathcal{M}'} f(\mathrm{id}_{\mathcal{M}}) \qquad \text{and} \qquad f(x_1) \circ_{\mathcal{M}'} f(x_2) \leq_{\mathcal{M}'} f(x_1 \circ_{\mathcal{M}} x_2).$$

To see in elementary terms why lax homomorphisms are natural, note that if we consider the inequality $f(x_1 \circ_{\mathcal{M}} x_2) \leq_{\mathcal{M}'} f(x_1) \circ_{\mathcal{M}'} f(x_2)$ and imagine what its 0-ary version should be, we obtain $f(\mathrm{id}_{\mathcal{M}}) \leq \mathrm{id}_{\mathcal{M}'}$.

A lazy homomorphism sits in-between lax and oplax, combining the first axiom of oplax with the second axiom of lax. This is new.

We would like to give some intuition, in elementary terms, about why *lazy* homomorphisms are interesting.

[†] That is, specialising laxness to the case of a locally posetal 2-category on one object, a.k.a. partially ordered monoid.

First, an informal computational intuition: Read a partially ordered monoid or group as a set of operations, which can be composed. Think of the partial order as describing 'execution' of these operations—so $z \leq x \circ y$ means "if we combine x with y and execute then we obtain z". Then we can give the lazy inequalities the following informal reading:

— $\mathrm{id}_{\mathcal{M}'} \leq_{\mathcal{M}'} f(\mathrm{id}_{\mathcal{M}})$ means *if we apply f to the identity and execute, we get the identity.*
— $f(x \circ_{\mathcal{M}} y) \leq_{\mathcal{M}'} f(x) \circ_{\mathcal{M}'} f(y)$ means *if we apply f to x and apply f to y and combine, then if we execute we can obtain the result of combining x and y in $\mathcal{M}$ first, and then applying f.*

We call this *lazy* by analogy with lazy evaluation in the λ-calculus.

Second, a mathematical intuition: We want $\mathcal{M}$ to inject into the 'imaginary' group $\mathcal{M}^{\pm}$ which augments $\mathcal{M}$ with 'imaginary' inverses (formal definition in Subsection 3.1). So let us consider a specific example and see what is required to make this work.

Let $\mathcal{M}$ be the one-element monoid $\{*\}$ with monoid composition $* \circ * = *$. Then $\mathcal{M}^{\pm}$ is isomorphic to the integers under addition: the unit is $()$; the number 1 corresponds to $+*$; the number 2 corresponds to $+*+*$; the number -1 corresponds to $-*$, and so on. The function taking $*$ to 1 is lazy because $0 \leq 1$ and $1 \leq 2$. This function is not lax, because $1 \not\leq 0$, and not oplax because $2 \not\leq 1$, but it is lazy.

So, with lazy homomorphisms the integers $\mathbb{Z}$ considered as a group with their natural arithmetic ordering are exhibited as the free object over a single generator.

Interestingly, our proof of injectivity hinges on studying *index functions* $\iota(\text{-})_m$ from $\mathcal{M}^{\pm}$ to $\mathbb{Z}$. See Definition 3.12. So the meat of our proof—the technical action happens between Definition 3.12 and Lemma 3.16—hinges on a study of the structure of the functions and objects making up the following commuting square:

$$\begin{array}{ccc} \mathcal{M} & \xrightarrow{\eta} & \mathcal{M}^{\pm} \\ {\scriptstyle !\text{ terminal map}}\downarrow & & \downarrow{\scriptstyle \iota(\text{-})} \\ \{*\} & \xrightarrow{*\mapsto 1} & \{*\}^{\pm} = \mathbb{Z} \end{array}$$

The reader should be careful: though the total index function $\iota(\text{-})$ (a 'sum of exponents' function) is an arrow in PoMonZ, the proof uses measure functions $\iota(\text{-})_m$ that are not arrows in this category. So the abstract nonsense above is useful for organising the material, but the proofs themselves cannot be conveniently phrased purely in categorical terms; we use combinatorial arguments on concrete constructions.

For reference: $\mathcal{M}^{\pm}$ is defined in Definition 3.3; η is defined in Theorem 3.30; the proof that η is mono is Theorem 3.22; and commutativity is an easy calculation. With respect to later notation, we have abused notation in the diagram above and written $\mathcal{M}^{\pm}$ for $(\mathcal{M}^{\pm})^{\mathcal{A}}$ (Definition 3.29).

3. The imaginary group

Our construction is the simplest possible; given a monoid $\mathcal{M}$ we take strings of elements-with-a-polarity $+$ or $-$ and quotient such that $+m-m$ becomes equal to the empty string.

We immediately obtain an *imaginary* group $\mathcal{M}^{\pm}$. It is the free group over $\mathcal{M}$ in the category of lazy monoids, to be precise. This is Definition 3.3 (the hard part is not constructing it, but proving it is not trivial; see the conservative extension results below).

For the following step we need a little general theory about partial orders on groups. It is a fact that a partial order can be generated by a normal submonoid. This turns out to be useful: see Proposition 3.7.

We then define a preorder on $\mathcal{M}^{\pm}$ in Figure 1, in submonoid style. The order is sound by construction; the rule (**leq**) makes sure that if $n \leq m$ in $\mathcal{M}$ then $+n \leq +m$ in $\mathcal{M}^{\pm}$.

The hard work is *injectivity* and *conservative extension* results, that the preorder is a partial order and is not 'too big'—we do not want $+n \leq +m$ to hold because $+n = +m$ always, in the spirit of Example 1.1. The main technical lemma is Proposition 3.19, and the main results are Theorems 3.22, 3.24 and 3.25.

Once we have this, it is a matter of fairly routine calculations to convert this into the adjunction promised in the Introduction, culminating with Theorem 3.30.

3.1. *The free group*

This subsection describes an easy construction. Given $\mathcal{M}$ we consider $|\mathcal{M}|$ and build the free group generated by this set.

Definition 3.1. Given a set X write $X^{\pm}$*string* for the set inductively defined by:

$$s ::= () \mid s{+}x \mid s{-}x \qquad (x \in X)$$

Call $()$ the **empty string**. Write string concatenation as $s{\cdot}t$. We may elide the initial $()$, so for instance we may write $+m$ as shorthand for $()+m$.

So if $X = \{1, 2, 3, \dots\}$ then intuitively $X^{\pm}$*string* is the set of lists of nonzero integers.

Definition 3.2. Define $\sim$ to be the least equivalence relation on $|\mathcal{M}|^{\pm}$*string* such that:

1 $()+m-m \sim () \sim ()-m+m$.
2 If $s \sim s'$ and $t \sim t'$ then $s{\cdot}t \sim s'{\cdot}t'$.

Definition 3.3. Given a monoid $\mathcal{M}$ define the **free group** $\mathcal{M}^{\pm}$ over $|\mathcal{M}|$ as follows:
— The carrier set is $|\mathcal{M}|^{\pm}$*string* quotiented by $\sim$.
— Group composition is given by concatenation of representatives.
We may write m for both $m \in |\mathcal{M}|$ and for the $\sim$-equivalence class of $() + m$.

Lemma 3.4. *$\mathcal{M}^{\pm}$ is indeed a group, with the natural composition, identity, and inverse.*

Example 3.5.

1 Suppose $\mathcal{M}$ is the monoid of natural numbers $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ under addition (so 0 is the identity). Then $\mathcal{M}^{\pm}$ is *not* isomorphic to the integers. For instance, $+3{+}4 \in |\mathcal{M}^{\pm}|$ is not equal to $+7$. We return to this in a mathematically richer context, later on in Remark 3.18.

2 Suppose $\mathcal{M}$ is the one-element monoid with underlying set $\{*\}$ and monoid composition $* \circ * = *$ (we also saw this in Subsection 2.2). Then $\mathcal{M}^{\pm}$ is isomorphic to the integers under addition: the unit is $()$; the number 1 corresponds to $+*$; the number 2 corresponds to $+*+*$; the number -1 corresponds to $-*$, and so on.
So $\mathcal{M}^{\pm}$ is *resource sensitive*. Note, however, that resources can be destroyed if we interpret $\geq$ as 'entailment'; $+*+* \geq +*$.[‡]

3.2. *Positive cones*

This subsection develops some more standard theory. There is a bijection between normal submonoids of $\mathcal{G}$ and compatible partial orders. We only need one direction of this for the proofs to follow; see [Glass, 1999, Subsection 1.1] for more details.

Definition 3.6.

— Suppose $\mathcal{G} = (|\mathcal{G}|, \circ, ^{-1})$ is a group. Call $X \subseteq |\mathcal{G}|$ **normal** when $x \in X$ and $y \in |\mathcal{G}|$ implies $y \circ x \circ y^{-1} \in X$.
— Suppose $\mathcal{M}$ is a monoid. Call $X \subseteq |\mathcal{M}|$ a **submonoid** when $\mathrm{id} \in X$ and when $x \in X$ and $y \in X$ implies $x \circ y \in X$.

Suppose $P \subseteq |\mathcal{G}|$ is a normal submonoid of a group $\mathcal{G}$. Then write $x \leq_P y$ when $y \circ x^{-1} \in P$.

Proposition 3.7. *Suppose $\mathcal{G} = (|\mathcal{G}|, \circ, ^{-1})$ is a group and $P \subseteq |\mathcal{G}|$ is a normal submonoid of $\mathcal{G}$. Then $\leq_P$ from Definition 3.6 is a compatible preorder on $|\mathcal{G}|$.*[§]

Proof. We consider each property in turn:

— By assumption $\mathrm{id} \in P$; it follows that $m \leq_P m$ and so $\leq_P$ is reflexive.
— Suppose $x \leq_P y$ and $y \leq_P z$. This means that $y \circ x^{-1} \in P$ and $z \circ y^{-1} \in P$. Since P is a monoid also $z \circ x^{-1} \in P$. It follows that $\leq_P$ is transitive.
— Suppose $x \leq_P y$. This means that $y \circ x^{-1} \in P$. To check that $\leq_P$ is compatible we need to verify that for any $z \in |\mathcal{G}|$, $z \circ x \leq_P z \circ y$ and $x \circ z \leq_P y \circ z$.
Since P is a normal subgroup also $z \circ y \circ x^{-1} \circ z^{-1} \in P$, and this means that $z \circ x \leq_P z \circ y$.
Also, $y \circ x^{-1} = y \circ z \circ z^{-1} \circ x^{-1}$, so that $x \circ z \leq_P y \circ z$.

□

[‡] In logical terminology, we are observing that we have contraction on the right but not on the left.
Contraction in logic is a rule that allows us to convert from $A \wedge A$ to A. If A represents an assertion and $\wedge$ represents logical conjunction, then this is immediate. If A represents a pound coin and $\wedge$ represents addition, then this may not be so.
'Contraction on the left' means we may go from A to $A \wedge A$ (like truth, or taxes) whereas 'contraction on the right' means we may go from $A \wedge A$ to A (like truth, or our bank account). More on this is in the field of *substructural logics*, and the interested reader is referred elsewhere [Restall, 1999].

[§] Recall that a *preorder* is reflexive and transitive, but not necessarily antisymmetric. A *partial order* is an antisymmetric preorder.

$$\frac{(n \leq_{\mathcal{M}} m)}{+m{-}n \in P^+}\ (\mathbf{leq}) \qquad \frac{(m_1, m_2 \in |\mathcal{M}|)}{+m_1{+}m_2{-}(m_1 \circ_{\mathcal{M}} m_2) \in P^+}\ (\mathbf{lazy})$$

$$\frac{p \in P^+ \quad (g \in \mathcal{M}^{\pm})}{g^{\text{-}1} \circ_{\mathcal{M}^{\pm}} p \circ_{\mathcal{M}^{\pm}} g \in P^+}\ (\mathbf{norm}) \qquad \frac{p \in P^+ \quad p' \in P^+}{p \circ_{\mathcal{M}^{\pm}} p' \in P^+}\ (\mathbf{mon})$$

Fig. 1: Rules determining the partial order on $\mathcal{M}^{\pm}$

3.3. *The partial order*

Now the non-standard part begins. Fix a partially ordered monoid $(\mathcal{M}, \leq_{\mathcal{M}})$. In Definition 3.3 we saw how to construct the free group $\mathcal{M}^{\pm}$ generated by $|\mathcal{M}|$.

We show how to extend $\leq_{\mathcal{M}}$ to a preorder $\leq_{\mathcal{M}^{\pm}}$ on $\mathcal{M}^{\pm}$. By Proposition 3.7 it suffices to exhibit a normal submonoid of $\mathcal{M}^{\pm}$. In fact the preorder is also a partial order, but proving this takes more work: the main technical result is Proposition 3.19 and this quickly yields Theorem 3.22.

Definition 3.8. Define P^+ to be the least subset of $|\mathcal{M}^{\pm}|$ closed under rules (**leq**), (**lazy**), (**norm**), and (**mon**) from Figure 1. By rules (**norm**) and (**mon**) this is a normal submonoid of $\mathcal{M}^{\pm}$ and so by Proposition 3.7 P^+ determines a compatible preorder on $|\mathcal{M}^{\pm}|$. We write this $\leq_{\mathcal{M}^{\pm}}$.

Remark 3.9. A few words on the rules in Figure 1.

— (**leq**) expresses that $n \leq_{\mathcal{M}} m$ implies $n \leq_{\mathcal{M}^{\pm}} m$.
— (**lazy**) expresses that $+(m_1 \circ_{\mathcal{M}} m_2) \leq_{\mathcal{M}^{\pm}} +m_1{+}m_2$.
— (**norm**) expresses that if $p \leq_{\mathcal{M}^{\pm}} q$ then $p \circ_{\mathcal{M}^{\pm}} r \leq_{\mathcal{M}^{\pm}} q \circ_{\mathcal{M}^{\pm}} r$ (left-compatibility is automatic by construction; see Proposition 3.7).
— (**mon**) expresses transitivity of $\leq_{\mathcal{M}^{\pm}}$.

The two base cases are (**leq**) and (**lazy**). Of these, rule (**leq**) introduces a string which has index 0; and rule (**lazy**) introduces a string which has index 1.

Lemma 3.10.

1 *If* $n \leq_{\mathcal{M}} m$ *then* $+n \leq_{\mathcal{M}^{\pm}} +m$.
2 *If* $n \leq_{\mathcal{M}} m_1 \circ_{\mathcal{M}} m_2$ *then* $+n \leq_{\mathcal{M}^{\pm}} +m_1{+}m_2$.

Proof. The first part is direct from (**leq**) and Definition 3.6. The second part follows using (**lazy**). □

Converse implications also hold but the proofs are harder: see Theorems 3.24 and 3.25.

Now (**lazy**) in Figure 1 corresponds to the condition $f(x_1 \circ_{\mathcal{M}'} x_2) \leq_{\mathcal{M}'} f(x_1) \circ_{\mathcal{M}'} f(x_2)$ from Definition 2.3. What corresponds to the condition $\mathrm{id}_{\mathcal{M}'} \leq_{\mathcal{M}'} f(\mathrm{id}_{\mathcal{M}})$ from Definition 2.3? In fact, this is derivable:

Corollary 3.11. $() \leq_{\mathcal{M}^{\pm}} +\mathrm{id}_{\mathcal{M}}$. *(Recall from Subsection 3.1 that* $()$ *is the identity element of* $\mathcal{M}^{\pm}$*.)*

Proof. By part 2 of Lemma 3.10 $+\mathrm{id}_{\mathcal{M}} \leq_{\mathcal{M}^{\pm}} +\mathrm{id}_{\mathcal{M}}+\mathrm{id}_{\mathcal{M}}$. We apply $-\mathrm{id}_{\mathcal{M}}$ to both sides. □

3.4. *Technical lemmas*

This is where the maths begins to become non-trivial. In Definition 3.8 we defined a relation $\leq_{\mathcal{M}^{\pm}}$ on $|\mathcal{M}^{\pm}|$. It is easy to see that this is reflexive and transitive; but is it antisymmetric? In other words, is $\leq_{\mathcal{M}^{\pm}}$ a partial order as the caption to Figure 1 suggests, and not just a preorder? Also, is $\leq_{\mathcal{M}^{\pm}}$ a conservative extension of $\leq_{\mathcal{M}}$ or does it introduce spurious relations that should not be there?

This subsection develops the technical machinery we need to answer these questions. These lemmas are then used to give the questions concise answers, in Subsection 3.5.

Definition 3.12. Suppose $m \in |\mathcal{M}|$. Define **index functions** $\iota(s)_m$ and $\iota(s)$ on $|\mathcal{M}|^{\pm}$*string* as follows:

$$\begin{array}{lll} \iota(())_m = 0 & \iota(s{+}m')_m = \iota(s)_m \ (m'{\neq}m) & \iota(s{-}m')_m = \iota(s)_m \ (m'{\neq}m) \\ \iota(s{-}m)_m = \iota(s)_m{-}1 & \iota(s{+}m)_m = \iota(s)_m{+}1 & \\ \iota(s) = \Sigma_{m\in|\mathcal{M}|}\iota(s)_m & & \end{array}$$

In words, we can say that $\iota(s)$ *sums exponents*.

Lemma 3.13. *The index functions are well-defined as functions on $p \in |\mathcal{M}^{\pm}|$. That is, if $p \in |\mathcal{M}^{\pm}|$ then for every $s, s' \in p$ and every m it is the case that $\iota(s)_m = \iota(s')_m$ and (therefore) $\iota(s) = \iota(s')$.*

Proof. We consider each of the rules in Definition 3.2 and see that they conserve index. □

Lemma 3.14. *If $() \leq_{\mathcal{M}^{\pm}} p$ and $\iota(p) = 0$ then every proof that $p \in P^+$, does not mention* (**lazy**).

Proof. We note by arithmetic that (**leq**) generates an element with index 0, (**lazy**) increases the index by 1, and (**norm**) and (**mon**) leave the index unchanged. The result follows. □

Corollary 3.15. *If $() \leq_{\mathcal{M}^{\pm}} p$ and $\iota(p)_m = 0$ for all m, then $p = ()$.*

Proof. We first use Lemma 3.14 to see that the derivation of $() \leq_{\mathcal{M}^{\pm}} p$ cannot use rule (**lazy**). The rules (**norm**) and (**mon**) do not affect index at all.

Let n be $\leq_{\mathcal{M}}$-minimal in p. This n must have been introduced by (**leq**) for some $n \leq_{\mathcal{M}} m$. Since n is $\leq_{\mathcal{M}}$-minimal in p there is no other instance of (**leq**) for $n' \leq_{\mathcal{M}} n$. By assumption $\iota(p)_n = 0$, so it must be that $n = m$. The result follows. □

Lemma 3.16.

1 *If $p \in P^+$ then $\iota(p) \geq 0$.*
2 *If $p \leq_{\mathcal{M}^{\pm}} q$ then $\iota(p) \leq \iota(q)$.*

Proof. The first part is by a routine induction on the rules in Figure 1.

For the second part, suppose $p \leq_{\mathcal{M}^\pm} q$. Then $() \leq_{\mathcal{M}^\pm} p^{-1} \circ_{\mathcal{M}^\pm} q$. By part 1 of this result, $\iota(p^{-1} \circ_{\mathcal{M}^\pm} q) \geq 0$. It is a fact that $\iota(p^{-1} \circ_{\mathcal{M}^\pm} q) = -\iota(p) + \iota(q)$. The result follows. □

Corollary 3.17. *If* $+m_1 \ldots +m_k \leq_{\mathcal{M}^\pm} +n_1 \ldots +n_l$ *and* $+n_1 \ldots +n_l \leq_{\mathcal{M}^\pm} +m_1 \ldots +m_k$ *then* $k = l$.

Proof. $\iota(+m_1 \ldots +m_k) = k$ and $\iota(+n_1 \ldots +n_l) = l$. We use part 2 of Lemma 3.16. □

Remark 3.18. Recall part 1 of Example 3.5 and take $\mathcal{M}$ to be the monoid of natural numbers $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ under addition (so 0 is the identity). Then by Corollary 3.17 $+3+4 \not\leq_{\mathcal{M}^\pm} +7$, since $\iota(+3+4) = 2$ and $\iota(+7) = 1$ and $2 \not\leq 1$.

In fact, a much stronger result than Corollary 3.17 can be obtained, but the proof is harder. See Corollary 3.23.

Proposition 3.19 might seem a little formidable, but in fact it only expresses some common sense: Suppose the index of p is 0, so that for every $+m$ in p there is a corresponding $-n$. Suppose there are more $+m$s in p than $-m$s, so that there is a surfeit of ms in p and $\iota(p)_m > 0$. In order for everything to balance out, there must be some n such that $\iota(p)_n < 0$—that is, such that there is a deficit of ns—and since p is a finite structure we can find an $\leq_{\mathcal{M}}$-minimal such n in p. This is what Proposition 3.19 asserts, and this property turns out to be key.

Proposition 3.19. *Suppose* $p \in P^+$ *and* $\iota(p) = 0$. *Suppose* $\iota(p)_m > 0$. *Then there exists some* $\leq_{\mathcal{M}}$*-minimal* $n \leq_{\mathcal{M}} m$ *such that* $\iota(p)_n < 0$.

Proof. By induction on the derivation that $p \in P^+$. We consider each rule in turn, in a long but essentially routine analysis of cases ((**mon**) is arguably the more interesting, and certainly the most complex, case).

— *The case of* (**leq**). Then $n \leq_{\mathcal{M}} m$, but we still need to show that $\iota(p)_m > 0$ and $\iota(p)_n < 0$. By assumption $\iota(+m-n)_m > 0$. This means that $m \neq n$. It follows by arithmetic that $\iota(+m-n)_n < 0$.

— *The case of* (**lazy**). If $\iota(+m_1+m_2-(m_1 \circ_{\mathcal{M}} m_2)) = 0$ then either $m_1 = m_2 = m = \mathrm{id}$ or $m_1 = m \neq \mathrm{id}$ and $m_2 = \mathrm{id}$ or $m_2 = m \neq \mathrm{id}$ and $m_1 = \mathrm{id}$. All cases are impossible because we assumed $\iota(p)_m > 0$ (for some m).

— *The case of* (**norm**). From the fact that $\iota(p)_m = \iota(g^{-1} \circ_{\mathcal{M}^\pm} p \circ_{\mathcal{M}^\pm} g)_m$.

— *The case of* (**mon**). Suppose $\iota(p \circ_{\mathcal{M}^\pm} p')_m > 0$. There are now two cases:

- *The case* $\iota(p)_m > 0$. By inductive hypothesis there exists a $\leq_{\mathcal{M}}$-minimal $n \leq_{\mathcal{M}} m$ such that $\iota(p)_n < 0$. There are now two sub-cases:
  - *The case* $\iota(p')_n < -\iota(p)_n$. In this case $\iota(p \circ_{\mathcal{M}^\pm} p')_n < 0$ and we are done.
  - *The case* $\iota(p')_n \geq -\iota(p)_n$. In this case, in particular, $\iota(p')_n > 0$. So there exists some $\leq_{\mathcal{M}}$-minimal $n' \leq_{\mathcal{M}} n$ such that $\iota(p')_{n'} < 0$. By assumption n was minimal, so $\iota(p)_n = 0$. It follows that $\iota(p \circ_{\mathcal{M}^\pm} p')_{n'} < 0$. By transitivity $n' \leq_{\mathcal{M}} m$ and we are done.
- *The case* $\iota(p)_m \leq 0 \ldots$ is symmetric with the case $\iota(p')_n < -\iota(p)_n$.

□

Proposition 3.19 has a natural dual:

Proposition 3.20. *Suppose $p \in P^+$ and $\iota(p) = 0$. Suppose $\iota(p)_m < 0$. Then there exists a $\leq_{\mathcal{M}}$-maximal $n \geq_{\mathcal{M}} m$ such that $\iota(p)_n > 0$.*

Proof. We take $\mathcal{M}$ and p and consider $\mathcal{M}'$ which is $\mathcal{M}$ but we invert $\leq_{\mathcal{M}}$ (we omit a formal definition), and p^{-1}. Thus, $\iota(p)_m < 0$ becomes $\iota(p^{-1})_m > 0$ and $n \geq_{\mathcal{M}} m$ becomes $n \leq_{\mathcal{M}'} m$, and '$\leq_{\mathcal{M}}$-maximal' becomes '$\leq_{\mathcal{M}'}$-minimal'. The result is then just Proposition 3.19. □

Corollary 3.21. *Suppose $p \in P^+$ and $\iota(p) = 1$. Then there exist $n, m_1, m_2 \in |\mathcal{M}|$ such that $n \leq_{\mathcal{M}} m_1 \circ_{\mathcal{M}} m_2$ and $\iota(p)_n \leq -1$ and $\iota(p)_{m_1} \geq 1$ and $\iota(p)_{m_2} \geq 1$.*

Proof. By induction on the derivation that $p \in P^+$. We consider each rule in turn:

— *The case of* (**leq**) is not relevant since $\iota(+m-n) = 0$.
— *The case of* (**lazy**). By construction, since $m_1 \circ_{\mathcal{M}} m_2 \leq_{\mathcal{M}} m_1 \circ_{\mathcal{M}} m_2$.
— *The case of* (**norm**). From the fact that $\iota(p)_m = \iota(g^{-1} \circ_{\mathcal{M}^\pm} p \circ_{\mathcal{M}^\pm} g)_m$.
— *The case of* (**mon**). Suppose $\iota(p \circ_{\mathcal{M}^\pm} p') = 1$. Then by part 1 of Lemma 3.16 precisely one of $\iota(p) = 1$ and $\iota(p') = 0$ or $\iota(p') = 1$ and $\iota(p) = 0$ must hold.
Suppose $\iota(p) = 1$ and $\iota(p') = 0$; the other case is symmetric and we will not consider it. By inductive hypothesis we obtain n, m_1, and m_2 as required, in p.
Now if $\iota(p')_n \geq -\iota(p)_n$ then we use Proposition 3.19 to obtain an $n' \leq_{\mathcal{M}} n$ such that $n' \leq_{\mathcal{M}} m_1 \circ_{\mathcal{M}} m_2$ and then use transitivity of $\leq_{\mathcal{M}}$.
Similarly if $\iota(p')_{m_1} \leq -\iota(p)_{m_1}$ then we use Proposition 3.20 and compatibility of the partial order with $\circ_{\mathcal{M}}$. Similarly for m_2. The result follows.

□

3.5. *The preorder is a partial order and the natural map is injective*

Theorem 3.22. *If $p \leq_{\mathcal{M}^\pm} q$ and $q \leq_{\mathcal{M}^\pm} p$ then $p = q$.*

Proof. Suppose $p \leq_{\mathcal{M}^\pm} q$ and $q \leq_{\mathcal{M}^\pm} p$. Write $P = q \circ_{\mathcal{M}^\pm} p^{-1}$ and $Q = p \circ_{\mathcal{M}^\pm} q^{-1}$. Thus $() \leq_{\mathcal{M}^\pm} P$ and $() \leq_{\mathcal{M}^\pm} Q$. Also by part 2 of Lemma 3.16 $\iota(p) = \iota(q)$ and so $\iota(P) = 0$.

Now if $\iota(P)_m = 0$ for all m then using Corollary 3.15 it follows that $p = q$ and we are done.

So suppose $\iota(P)_{m'} \neq 0$ for some m'. We will deduce a contradiction. If $\iota(P)_{m'} \neq 0$ for some m' then $\iota(P)_m > 0$ for some m. By Proposition 3.19 $\iota(P)_n < 0$ for some $n \leq_{\mathcal{M}} m$ and n is $\leq_{\mathcal{M}}$-minimal such amongst monoid elements mentioned in P. But then $\iota(Q)_n > 0$ and by Proposition 3.19 again $\iota(Q)_{m''} < 0$ for some $m'' \leq_{\mathcal{M}} n$. This contradicts minimality of n. □

We immediately apply Theorem 3.22 to strengthen Corollary 3.17:

Corollary 3.23. *If* $+m_1 \ldots +m_k \leq_{\mathcal{M}^\pm} +n_1 \ldots +n_l$ *and* $+n_1 \ldots +n_l \leq_{\mathcal{M}^\pm} +m_1 \ldots +m_k$ *then* $k = l$ *and* $m_i = n_i$ *for every* $1 \leq i \leq k$.

Proof. By Theorem 3.22 $+m_1 \ldots +m_k = +n_1 \ldots +n_l$. The result follows from Definition 3.3. □

Theorem 3.24 is a conservative extension result expressing that $\mathcal{M}$ injects into $\mathcal{M}^\pm$ considered as a partially ordered monoid:

Theorem 3.24. $+n \leq_{\mathcal{M}^\pm} +m$ *if and only if* $n \leq_{\mathcal{M}} m$.

Proof. The right-to-left implication is part 1 of Lemma 3.10. For the left-to-right implication we reason as follows: If $m = n$ then $n \leq_{\mathcal{M}} m$ by reflexivity. So suppose $m \neq n$ and suppose $+n \leq_{\mathcal{M}^\pm} +m$.

By Definition 3.6 this means that $+m{-}n \in P^+$. We note that $\iota(+m{-}n) = 0$ and $\iota(+m{-}n)_m > 0$. Using Proposition 3.19 it follows that $n \leq_{\mathcal{M}} m$. □

$\mathcal{M}^\pm$ also does not introduce any new 'lazy' relations:

Theorem 3.25. $+n \leq_{\mathcal{M}^\pm} +m_1{+}m_2$ *if and only if* $n \leq_{\mathcal{M}} m_1 \circ_{\mathcal{M}} m_2$.

Proof. The right-to-left implication is part 2 of Lemma 3.10. The left-to-right implication follows using Corollary 3.21. □

3.6. *The functor from* PoMonZ *to* PoGrp

Definition 3.26. Define a functor imag : PoMonZ → PoGrp as follows:

— $\mathcal{M} \in$ PoMonZ maps to $(\mathcal{M}^\pm, \leq_{\mathcal{M}^\pm})$ (Definitions 3.3 and 3.8).

— An arrow $f : \mathcal{M} \to \mathcal{M}' \in$ PoMonZ acts elementwise on representatives of $\sim$-equivalence classes. In full:

$$\begin{aligned} \mathsf{imag}(f)() &= () \\ \mathsf{imag}(f)(s{+}m) &= \mathsf{imag}(f)(s){+}f(m) \\ \mathsf{imag}(f)(s{-}m) &= \mathsf{imag}(f)(s){-}f(m) \end{aligned}$$

Lemma 3.27. imag *from Definition 3.26 is well-defined and is a functor from* PoMonZ *to* PoGrp.

Proof. Since $\mathsf{imag}(f)$ acts elementwise, $\mathsf{imag}(f)(s{\cdot}s') = \mathsf{imag}(f)(s){\cdot}\mathsf{imag}(f)(s')$ and $\mathsf{imag}(f)() = ()$ by construction. Thus, $\mathsf{imag}(f)$ is a group homomorphism.

We need to check that $\mathsf{imag}(f) : \mathsf{imag}(\mathcal{M}) \to \mathsf{imag}(\mathcal{M}')$ is well-defined on $\sim$-equivalence classes. We sketch the two least trivial cases:

— $\mathsf{imag}() = \mathrm{id}_{\mathcal{M}'} = f(\mathrm{id}_{\mathcal{M}}) = \mathsf{imag}(\mathrm{id}_{\mathcal{M}})$. (This is slightly interesting because it depends on our assumption that $f(\mathrm{id}_{\mathcal{M}}) = \mathrm{id}_{\mathcal{M}'}$.)

— $\mathsf{imag}(s{+}m{-}m) = \mathsf{imag}(s){+}f(m){-}f(m) \sim \mathsf{imag}(s)$, and similarly for $s{-}m{+}m$.

Finally, we must check that $\mathsf{imag}(f)$ is monotone. To do this it suffices to show that if $p \in P_{\mathcal{M}}$ then $\mathsf{imag}(p) \in P_{\mathcal{M}'}$. It suffices to check the following (note that the second case is not what one might immediately expect):

— *If* $n \leq_{\mathcal{M}} m$ *then* $f(n) \leq_{\mathcal{M}} f(m)$. This follows since f is monotone by assumption.
— *If* $m = m_1 \circ_{\mathcal{M}} m_2$ *then* $f(m) \leq_{\mathcal{M}'} f(m_1) \circ_{\mathcal{M}'} f(m_2)$. This follows since f is lazy.

□

Example 3.28. Consider the natural numbers $\{0, 1, 2, \dots\}$ with their usual ordering. This can be regarded as an object in PoMonZ, with 0 being the additive identity. Write this object just as $\mathbb{N}$. Consider the function $f : \mathbb{N} \to \mathbb{N}$ defined by

$$\begin{aligned} f(n) &= n + 1 \text{ if } n \neq 0 \\ f(0) &= 0. \end{aligned}$$

Then

$$\begin{aligned} f(n+m) &= n + m + 1 \\ &\leq n + 1 + m + 1 \\ &= f(n) + f(m), \end{aligned}$$

unless one or both of n and m are zero in which case $f(n+m) = f(n) + f(m)$. So f is an example of a monotone lazy homomorphism and is an arrow in PoMonZ from $\mathbb{N}$ to itself.

Then $\mathsf{imag}(\mathbb{N}) = \mathbb{N}^{\pm}$ is the free group on $\{0, 1, 2, \dots\}$ and $\mathsf{imag}(f)$ acts pointwise on strings so that for example

$$\mathsf{imag}(f)(+0+1+2) = +0+2+3.$$

This illustrates why in Definition 2.4 we took homomorphisms, not lazy homomorphisms, between the partially ordered groups; $\mathsf{imag}(f)$ is always a homomorphism, by construction.

3.7. *The forgetful functor from* PoGrp *to* PoMonZ

Definition 3.29. Define a functor $\mathsf{forg} : \mathsf{PoGrp} \to \mathsf{PoMonZ}$ as follows:

— $\mathcal{G} \in \mathsf{PoGrp}$ maps to $\mathcal{G}$ considered as a monoid; write this $\mathcal{G}^{\not A}$.
— An arrow $g : \mathcal{G} \to \mathcal{G}' \in \mathsf{PoGrp}$ maps to itself considered as a (lazy) monoid homomorphism.

Theorem 3.30. $\mathsf{imag} : \mathsf{PoMonZ} \to \mathsf{PoGrp}$ *is left adjoint to* $\mathsf{forg} : \mathsf{PoGrp} \to \mathsf{PoMonZ}$. *In symbols:*

$$\mathsf{imag} \dashv \mathsf{forg} : \mathsf{PoGrp} \to \mathsf{PoMonZ}$$

Proof. We exhibit the unit and counit of the adjunction.

The unit $\eta_{\mathcal{M}} : \mathcal{M} \to (\mathcal{M}^{\pm})^{\not A}$ is given by the function mapping $m \in |\mathcal{M}|$ to $+m \in |\mathcal{M}^{\pm}|$. By part 1 of Lemma 3.10 this is monotone. By part 2 of Lemma 3.10 and Corollary 3.11 it is a lazy homomorphism (Definition 2.3).

The counit $\epsilon_{\mathcal{G}} : (\mathcal{G}^{\pm})^{\not A} \to \mathcal{G}$ is given by the function inductively defined on represen-

tatives as follows:

$$\begin{aligned}\epsilon_{\mathcal{G}}() &= \mathrm{id}_{\mathcal{G}}\\ \epsilon_{\mathcal{G}}(s{+}m) &= \epsilon_{\mathcal{G}}(s) \circ_{\mathcal{G}} m\\ \epsilon_{\mathcal{G}}(s{-}m) &= \epsilon_{\mathcal{G}}(s) \circ_{\mathcal{G}} m^{-1}\end{aligned}$$

By construction this is a group homomorphism. We prove inductively on the definition of $\leq_{(\mathcal{G}\mathcal{A})^{\pm}}$ in Figure 1 that the map is monotone, that is, if $s \leq_{(\mathcal{G}\mathcal{A})^{\pm}} s'$ then $\epsilon_{\mathcal{G}}(s) \leq_{\mathcal{G}} \epsilon_{\mathcal{G}}(s')$:

— *Rule* (**leq**). Suppose $+n \leq_{(\mathcal{G}\mathcal{A})^{\pm}} +m$ because $n \leq_{\mathcal{G}\mathcal{A}} m$. Then $n \leq_{\mathcal{G}} m$ and we are done.

— *Rule* (**lazy**). Suppose $+m \leq_{(\mathcal{G}\mathcal{A})^{\pm}} +m_1{+}m_2$ because $m, m_1, m_2 \in |\mathcal{G}^{\pm}|$ and $m = m_1 \circ_{\mathcal{G}\mathcal{A}} m_2$. Then by construction

$$\epsilon_{\mathcal{G}}(+m) = \epsilon_{\mathcal{G}}(+m_1{+}m_2),$$

and we are done.

— *Rules* (**norm**) *and* (**mon**) follow by inductive hypothesis and by compatibility and transitivity of $\leq_{\mathcal{G}}$.

It is not hard to see that the adjunction maps

— $g : \mathcal{M}^{\pm} \to \mathcal{G} \in \mathsf{PoGrp}$ to $\bar{g} : \mathcal{M} \to \mathcal{G}^{\mathcal{A}}$ which takes m to $g(+m)$, and

— $f : \mathcal{M} \to \mathcal{G}^{\mathcal{A}} \in \mathsf{PoMon}$ to $\bar{f} : \mathcal{M}^{\pm} \to \mathcal{G}$ which acts on generators by taking $+m$ to $f(m)$ and $-m$ to $f(m)^{-1}$.

Verifying naturality is routine. □

Remark 3.31. In Definition 3.12 we defined the *measure function* $\iota(\text{-})$ from $|\mathcal{M}^{\pm}|$ to integers $\mathbb{Z}$ considered as a group under addition, for each partially ordered monoid $\mathcal{M}$. It is easy to check that $\iota(\text{-})$ is a monotone homomorphism of groups so by Theorem 3.30 it must correspond to a unique monotone lazy homomorphism from $\mathcal{M}$ to $\mathbb{Z}^{\mathcal{A}}$ (i.e. to $\mathbb{Z}$ considered as a monoid under addition).

It is not hard to see that this monotone lazy homomorphism is the trivial function mapping every $m \in |\mathcal{M}|$ to $1 \in \mathbb{Z}$. The laziness conditions from Definition 2.3 then correspond to the arithmetic inequalities $0 \leq 1$ and $1 \leq 2$.

One could consider other measure functions coming from different functions out of $\mathcal{M}$; discovering whether any of them are useful, is future research.

From the point of view of these general categorical observations, the injectivity and conservative extension results Theorems 3.22, 3.24 and 3.25 are proved by a detailed analysis of the fine structure of this measure function; we decompose it into $\iota(\text{-})_m$ for each m—which is not monotone and so is not an arrow in PoGrp—and then reason by a succession of counting arguments which culminate with Proposition 3.19.

4. Commutative and nonassociative generalisation

In this brief section we make a few observations about generalising the proofs.

4.1. *Commutative and other generalisations*

The key technical lemmas in Subsection 3.4 are by counting arguments, where the counting is performed by the index function of Definition 3.12. Because of this, it is not hard to verify that we can re-play all the proofs for the special cases where $\mathcal{M}$ and $\mathcal{M}^{\pm}$ are taken to be abelian.

That is, even if $x \circ_{\mathcal{M}} y = y \circ_{\mathcal{M}} x$ for all $x, y \in |\mathcal{M}|$, by our construction in Definition 3.2 it is not the case that $+x+y \sim +y+x$—what is true is that $+x+y \geq_{\mathcal{M}^{\pm}} +(x \circ_{\mathcal{M}} y) = +(y \circ_{\mathcal{M}} x)$. However, none of the proofs depend on this; the proofs we have seen only count elements, and we can rearrange elements however we like. So if $\mathcal{M}$ is commutative we could make $\mathcal{M}^{\pm}$ be commutative too, just by extending $\sim$ in Definition 3.2. Then we would obtain an adjunction between abelian monoids and abelian groups ((**mon**) in Figure 1 would become irrelevant).

Similarly, there is nothing in our proofs that depends on associativity of $\mathcal{M}$, or even on the existence of a unit (we would drop the first laziness condition Definition 2.3).

So it is not hard to generalise or specialise the results of this paper to e.g.

— abelian monoids (monoids in which $x \circ y = y \circ x$) and abelian groups—the theorems in Subsection 3.5 should survive unchanged except that Corollary 3.23 would have to take m_i and n_i up to some permutative rearrangement,
— semigroups (like monoids, but without a unit) and groups, so that we add both inverses *and* a unit,
— applicative structures (like monoids, but without a unit and without associativity) and applicative structures with inverses and a unit (but without associativity),

and so on. For instance elements of a Boolean algebra form a commutative monoid under $\wedge$; we can turn this into a commutative group.

In that sense, the proofs of this paper are robust: so long as none of the congruences we choose to put in or take out of Definition 3.2 affect the index function of Definition 3.12, then the proofs should survive.

4.2. *Generalisation to categories and groupoids*

It is not hard to add types to our results, i.e. to generalise from monoids to categories. We sketch the relevant definitions:

Definition 4.1. Call a category $\mathcal{C}$ **locally posetal** when its hom-sets are enriched to be partial orders, and this is consistent with arrow composition.[¶]

A **groupoid** is a category in which every arrow has an inverse.

A locally posetal category can be viewed as an instance of a *2-category* [Leinster, 2004, Cheng and Lauda, 2004] in which the hom-sets are posets.

Definition 4.2. A **lazy functor** on locally posetal categories $F : \mathcal{C} \to \mathcal{D}$ is a map from objects of $\mathcal{C}$ to objects of $\mathcal{D}$ and from hom-posets $\mathcal{C}(X, Y)$ to $\mathcal{D}(F(X), F(Y))$ such that:

[¶] So if $f' \leq f$ are two arrows in $\mathcal{C}(X, Y)$ and $g' \leq g$ are two arrows in $\mathcal{C}(Y, Z)$ then $f \circ g \leq f' \circ g'$ in $\mathcal{C}(X, Z)$.

— If $f' \leq f$ in $\mathcal{C}(X,Y)$ then $F(f') \leq F(f)$ in $\mathcal{D}(F(X),F(Y))$.
— $\mathrm{id}_{F(X)} \leq F(\mathrm{id}_X)$ for every $X \in Obj(\mathcal{C})$.
— $F(f \circ g) \leq F(f) \circ F(g)$ for every $f \in \mathcal{C}(X,Y)$ and $g \in \mathcal{C}(Y,Z)$.

Write PoCatZ and PoGrpoid for the categories of small locally posetal categories and lazy functors, and small locally posetal groupoids and functors between them, respectively.

Definition 4.3. Given a locally posetal category $\mathcal{C}$ define a locally posetal groupoid $\mathcal{C}^{\pm}$ as follows:

— Objects of $\mathcal{C}^{\pm}$ are objects of $\mathcal{C}$.
— An arrow is the quotiented well-typed[‖] formal sum of arrows from $\mathcal{C}$ just as in Definition 3.3, where if $f \in \mathcal{C}(X,Y)$ then $+f$ is given type $\mathcal{C}^{\pm}(X,Y)$ and $-f$ is given type $\mathcal{C}^{\pm}(Y,X)$.
— The partial order is defined on arrows just as in Figure 1 (thus: only on *well-typed* representatives).

With Definition 4.3 the adjunction of Subsection 3.7 extends to an adjunction between PoCatZ and PoGrpoid. The proof of injectivity is no different from the case of monoids; we just have to keep track of types. So for instance:

Definition 4.4. Define **index functions** $\iota(s)_m$ and $\iota(s)$ just as in Definition 3.12.

The well-typedness conditions are orthogonal to the counting arguments that follow, and it is easy to replay the proofs to obtain, for instance, the following:

Theorem 4.5.

— *$+f' \leq +f$ in $\mathcal{C}^{\pm}(X,Y)$ if and only if $f' \leq f$, for $f', f \in \mathcal{C}(X,Y)$.*
— *$+f \leq +g+h$ in $\mathcal{C}^{\pm}(X,Z)$ if and only if $f \leq g \circ h$ in $\mathcal{C}(X,Z)$, for $f \in \mathcal{C}(X,Z)$ and $g \in \mathcal{C}(X,Y)$ and $h \in \mathcal{C}(Y,Z)$.*

We leave it as an exercise to the interested reader to fill in the details of the relevant generalisations. Since our proofs are by counting arguments, the generalisation to the typed case introduces extra richness but no essential difficulties.

5. Sets with an action

We now extend the results of Section 3 to sets with a monoid action, showing how to extend a partially ordered set with an $\mathcal{M}$ monoid action to a partially ordered set with an $\mathcal{M}^{\pm}$ group action.

Examples of sets with a monoid action are common in computer science, including:

— Syntax acted on by a monoid of substitutions.
— Integers $\mathbb{Z}$ acted on by the monoid of functions from $\mathbb{Z}$ to $\mathbb{Z}$.

[‖] We just do the obvious thing here: well-typed means, for instance, that we only allow ourselves to write $+f+g$ when $f \in \mathcal{C}(X,Y)$ and $g \in \mathcal{C}(Y,Z)$. It is easy to see that e.g. $+f-f$ is well-typed and in $\mathcal{C}(X,X)$.

— A domain model of the λ-calculus D acted on by $D^D \cong D$ considered as a monoid under functional composition.

So, it is an interesting question to see how our results can be extended.

Subsection 5.1 contains the basic definitions, several key technical lemmas, and one important result (Theorem 5.9). Subsection 5.2 defines the relevant adjunctions between the categories of partially ordered sets with a compatible $\mathcal{M}$-action and lazy functions between them, and of partially ordered sets with an $\mathcal{M}^{\pm}$-action and functions between them.

The proofs are easier than in Section 3, but they are not trivial and do not follow by abstract nonsense. Certainly, if we can extend monoids to groups then we can extend sets acted on by monoids to sets acted on by groups in a standard way.

Just as in Section 3, what does *not* follow from abstract nonsense is injectivity results like Theorem 5.9 or Proposition 5.15, which are proved, just as in Section 3, by careful counting.

5.1. *Construction of the sets with an action*

Definition 5.1. Suppose $(\mathcal{M}, \leq_{\mathcal{M}}) \in \mathsf{PoMonZ}$. A **poset with an $\mathcal{M}$-action** $\mathcal{X}$ is a tuple $(|\mathcal{X}|, \cdot_{\mathcal{X}}, \leq_{\mathcal{X}})$ of a **carrier set** $|\mathcal{X}|$, a partial order $\leq_{\mathcal{X}}$, and an action $x \cdot_{\mathcal{X}} m$ such that:

— If $x \leq_{\mathcal{X}} x'$ and $m \leq_{\mathcal{M}} m'$ then $x \cdot_{\mathcal{X}} m \leq x' \cdot_{\mathcal{X}} m'$ (*the order on $\mathcal{X}$ is compatible with that on $\mathcal{M}$*).

— $x \cdot_{\mathcal{X}} \mathrm{id}_{\mathcal{M}} = x$ and $(x \cdot_{\mathcal{X}} m) \cdot_{\mathcal{M}} m' = x \cdot_{\mathcal{X}} (m \circ_{\mathcal{M}} m')$ (*the action is a monoid action*).

We define a **poset with a $\mathcal{G}$ action** similarly, for $(\mathcal{G}, \leq_{\mathcal{G}})$ a partially ordered group.

Suppose a poset with a compatible $\mathcal{M}$-action $\mathcal{X} = (|\mathcal{X}|, \cdot_{\mathcal{M}})$. We show how to extend this to a set with an $\mathcal{M}^{\pm}$-action.

Definition 5.2. Define $\leq$ to be the least transitive relation on $|\mathcal{X}| \times |\mathcal{M}^{\pm}|$ such that:

$$\frac{}{(x \cdot_{\mathcal{X}} m, g) \leq (x, +m \circ_{\mathcal{M}^{\pm}} g)}\ (\textbf{absorb}) \qquad \frac{x \leq_{\mathcal{X}} x' \quad g \leq_{\mathcal{M}^{\pm}} g'}{(x, g) \leq_{\mathcal{X}^{\pm}} (x', g')}\ (\textbf{compat})$$

$$\frac{(x, g) \leq (x', g') \quad (x', g') \leq (y, h)}{(x, g) \leq (y, h)}\ (\textbf{trans})$$

Lemma 5.3. *The ordering $\leq$ is* compatible, *in the sense that $(x, g) \leq (y, h)$ implies $(x, g \circ_{\mathcal{M}^{\pm}} g') \leq (y, h \circ_{\mathcal{M}^{\pm}} g')$.*

Proof. By an easy induction on the rules in Definition 5.2. □

Remark 5.4. By assumption $\leq_{\mathcal{X}}$ is reflexive and by construction so is $\leq_{\mathcal{X}^{\pm}}$, thus by (**compat**) $\leq$ is too. We still need to prove that $\leq_{\mathcal{X}^{\pm}}$ is antisymmetric, that is, we need to show that $\mathcal{X}^{\pm}$ is a partially ordered set and not only a preordered set. This we do next, culminating with Theorem 5.9.

Lemma 5.5. *If $(x,g) \leq (y,h)$ then there exist $m_1,\ldots,m_k \in |\mathcal{M}|$ such that $+m_1 \ldots +m_k \leq_{\mathcal{M}^\pm} h \circ_{\mathcal{M}^\pm} g^{-1}$.*

Proof. By an easy induction on the rules in Definition 5.2. □

Lemma 5.6. *If $(x,g) \leq (y,h)$ then $\iota(g) \leq \iota(h)$.*

Proof. By induction on Definition 5.2:

— *The case of* (**absorb**). It is a fact that $\iota(g) + 1 = \iota(m \circ_{\mathcal{M}^\pm} g)$.
— *The case of* (**compat**). If $g \leq_{\mathcal{M}^\pm} h$ then by part 2 of Lemma 3.16 $\iota(g) \leq \iota(h)$.
— *The case of* (**trans**). A fact of numbers.

□

Lemma 5.7. *If $(x,g) \leq (y,h)$ and $(y,h) \leq (z,g)$ then $g = h$.*

Proof. Suppose $(x,g) \leq (y,h)$ and $(y,h) \leq (z,g)$. By Lemma 5.6 $\iota(g) = \iota(h)$. Using this fact and Lemma 5.5 it follows that $g \leq_{\mathcal{M}^\pm} h$ and $h \leq_{\mathcal{M}^\pm} g$. Therefore by Theorem 3.22 $g = h$. □

Lemma 5.8. *If $(x,g) \leq (y,g)$ then $x \leq_{\mathcal{X}} y$.*

Proof. By induction on the rules in Definition 5.2.

— (**absorb**) cannot be used since $g = +m \circ_{\mathcal{M}^\pm} g$ is impossible since $\iota(g) \neq \iota(+m \circ_{\mathcal{M}^\pm} g) = 1 + \iota(g)$ and this contradicts part 2 of Lemma 3.16.
— The case of (**compat**) is immediate.
— For (**trans**), suppose $(x,g) \leq (x',g')$ and $(x',g') \leq (y,g)$. By Lemma 5.7 $g' = g$. We use the inductive hypothesis and transitivity of $\leq_{\mathcal{X}}$.

□

Theorem 5.9. *If $(x,g) \leq (y,h)$ and $(y,h) \leq (x,g)$ then $x = y$ and $g = h$.*

Proof. By Lemmas 5.7 and 5.8 and the fact that $\mathcal{X}$ and (by Theorem 3.22) $\mathcal{M}^\pm$ are partially ordered sets. □

5.2. *The natural adjunction*

Definition 5.10. Let $\mathcal{X}^\pm$ have underlying set $|\mathcal{X}| \times |\mathcal{M}^\pm|$ and action $\cdot_{\mathcal{X}^\pm}$ given by
$$(x,g) \cdot_{\mathcal{X}^\pm} h = (x, g \circ_{\mathcal{M}^\pm} h).$$

Definition 5.11. Suppose $\mathcal{M}$ is a partially ordered monoid. Write $\mathcal{M}$PoSetZ for the category such that:

— An object $\mathcal{X}$ is a poset with a compatible $\mathcal{M}$-action.
— An arrow $f : \mathcal{X} \to \mathcal{X}'$ is a function $f \in |\mathcal{X}| \to |\mathcal{X}'|$ such that
  – $x \leq_{\mathcal{X}} x'$ implies $f(x) \leq_{\mathcal{X}'} f(x')$.
  – $f(x \cdot_{\mathcal{X}} m) \leq_{\mathcal{X}'} f(x) \cdot_{\mathcal{X}'} m$.

Suppose $\mathcal{G}$ is a partially ordered group. Write $\mathcal{G}$PoSet for the category such that:

— An object $\mathcal{Y}$ is a poset with a compatible $\mathcal{G}$-action.
— An arrow $f : \mathcal{Y} \to \mathcal{Y}'$ is a function $f \in |\mathcal{Y}| \to |\mathcal{Y}'|$ such that:
  – $y \leq_{\mathcal{Y}} y'$ implies $f(y) \leq_{\mathcal{Y}'} f(y')$.
  – $f(y \cdot_{\mathcal{Y}} g) = f(y) \cdot_{\mathcal{Y}'} g$.

Definition 5.12. Let $\mathsf{imag} : \mathcal{M}\mathsf{PoSetZ} \to \mathcal{M}^{\pm}\mathsf{PoSet}$ have the following action:

— An object $\mathcal{X}$ maps to $\mathcal{X}^{\pm}$ (Definition 5.10).
— An arrow $f : \mathcal{X} \to \mathcal{X}'$ maps to $\mathsf{imag}(f) : \mathcal{X}^{\pm} \to (\mathcal{X}')^{\pm}$ which is defined by (x, g) maps to $(f(x), g)$.

Let $\mathsf{forg} : \mathcal{M}^{\pm}\mathsf{PoSet} \to \mathcal{M}\mathsf{PoSetZ}$ have the following action:

— An object $\mathcal{Y}$ (which is a set with an $\mathcal{M}^{\pm}$ group action) maps to $\mathcal{Y}$ considered as a set with an $\mathcal{M}$ action; write this $\mathcal{Y}_{\mathcal{M}}$.
— An arrow $f : \mathcal{Y} \to \mathcal{Y}' \in \mathcal{M}^{\pm}\mathsf{PoSet}$ maps to itself considered as a monoid homomorphism.

We consider an example of Definition 5.12 in Section 6.

Lemma 5.13. imag *from Definition 5.12 is a functor from* $\mathcal{M}\mathsf{PoSetZ}$ *to* $\mathcal{M}^{\pm}\mathsf{PoSet}$.

Proof. Suppose $f : \mathcal{X} \to \mathcal{Y} \in \mathcal{M}\mathsf{PoSetZ}$. We have two things to check:

— $\mathsf{imag}(f)$ *is monotone.* That is, we need to prove that if $(x, g) \leq (y, h)$ (Definition 5.2) in $|\mathcal{X}| \times |\mathcal{M}^{\pm}|$ then $(f(x), g) \leq (f(y), h)$ in $|\mathcal{X}'| \times |\mathcal{M}^{\pm}|$.
  We do this by induction on the derivation of $(x, g) \leq (y, h)$. The case of (**absorb**) follows from the assumption that $f(x \cdot_{\mathcal{X}} m) \leq_{\mathcal{X}'} f(x) \cdot_{\mathcal{X}'} m$ and using (**compat**); the case of (**compat**) is a structural fact; transitivity also follows.
— $\mathsf{imag}(f)$ *is a homomorphism.* That is, we need to prove that $(f(x), g) \cdot_{\mathcal{X}^{\pm}} h = (f(x), g \circ_{\mathcal{M}^{\pm}} h)$. But this holds direct from Definition 5.10.

□

Proposition 5.14. $\mathsf{imag} \dashv \mathsf{forg} : \mathcal{M}^{\pm}\mathsf{PoSet} \to \mathcal{M}\mathsf{PoSetZ}$.

Proof. The unit $\eta_{\mathcal{X}}$ maps $x \in |\mathcal{X}|$ to $(x, ()) \in |(\mathcal{X}^{\pm})_{\mathcal{M}}|$. We check that $\eta_{\mathcal{X}}(x \cdot_{\mathcal{X}} m) \leq \eta_{\mathcal{X}}(x) \cdot_{(\mathcal{X}^{\pm})_{\mathcal{M}}} m$:

$$\eta_{\mathcal{X}}(x) \cdot_{(\mathcal{X}^{\pm})_{\mathcal{M}}} m = (x, ()) \cdot_{\mathcal{X}^{\pm}} {+m} = (x, {+m}) \geq (x \cdot_{\mathcal{X}} m) = \eta_{\mathcal{X}}(x \cdot_{\mathcal{X}} m)$$

The counit $\epsilon_{\mathcal{Y}} : (\mathcal{Y}_{\mathcal{M}})^{\pm} \to \mathcal{Y}$ maps (x, g) to $x{\cdot_{\mathcal{Y}}}\epsilon_{\mathcal{M}^{\pm}}(g)$. We check that $\epsilon_{\mathcal{Y}}(x, g){\cdot_{\mathcal{Y}}}h = \epsilon_{\mathcal{Y}}(x, g \circ_{\mathcal{M}^{\pm}} h)$:

$$\epsilon_{\mathcal{Y}}(x, g){\cdot_{\mathcal{Y}}}h = (x{\cdot_{\mathcal{Y}}}g){\cdot_{\mathcal{Y}}}h = x{\cdot_{\mathcal{Y}}}(g \circ_{\mathcal{M}^{\pm}} h) = \epsilon_{\mathcal{Y}}(x, g \circ_{\mathcal{M}^{\pm}} h)$$

□

5.3. *More detailed analysis for the partial order*

In this subsection we go deeper into the structure of objects in $\mathcal{M}^{\pm}\mathsf{PoSet}$, asking the following question: if $x{\cdot}g \leq y{\cdot}h$ and $+m \leq g$, can $y{\cdot}h$ imitate this behaviour in some compatible sense?

This is Proposition 5.15—the reader who knows about process calculi can think of this loosely as a *simulation* result in the sense of e.g. [Milner, 1999, Definition 3.3].

Proposition 5.15. *Suppose $\mathcal{X}$ is a set with an $\mathcal{M}$-action. Suppose $(x, g) \leq (y, h)$. Then:*

1. *If $+m \leq_{\mathcal{M}^{\pm}} g$ then there exists $n \in |\mathcal{M}|$ such that $+n \leq_{\mathcal{M}^{\pm}} h$ and $x{\cdot_{\mathcal{M}}}m \leq_{\mathcal{X}} y{\cdot_{\mathcal{M}}}n$.*
2. *If $() \leq_{\mathcal{M}^{\pm}} g$ then either*
   - *$() \leq_{\mathcal{M}^{\pm}} h$ and $x \leq_{\mathcal{X}} y$, or*
   - *there exists $n \in |\mathcal{M}|$ such that $+n \leq_{\mathcal{M}^{\pm}} h$ and $x \leq_{\mathcal{X}} y{\cdot_{\mathcal{M}}}n$.*

Proof. For the first part we work by induction on the derivation of $\leq$.

- *The case of* (**absorb**). By (**absorb**) $(x{\cdot_{\mathcal{X}}}m, g) \leq (x, +m \circ_{\mathcal{M}^{\pm}} g)$. Suppose $+n \leq_{\mathcal{M}^{\pm}} g$. Then $+(m \circ_{\mathcal{M}} n) \leq_{\mathcal{M}^{\pm}} +m{+}n \leq_{\mathcal{M}^{\pm}} +m \circ_{\mathcal{M}^{\pm}} g$, and it is a fact that $(x{\cdot_{\mathcal{M}}}m){\cdot_{\mathcal{M}}}n = x{\cdot_{\mathcal{M}}}(m \circ_{\mathcal{M}} n)$.
  It follows that $(x{\cdot_{\mathcal{M}}}m){\cdot_{\mathcal{X}}}n \leq_{\mathcal{M}} x{\cdot_{\mathcal{M}}}(m \circ_{\mathcal{M}} n)$.
- *The case of* (**compat**). Suppose $(x, g) \leq (x', g')$ because $x \leq_{\mathcal{X}} x'$ and $g \leq_{\mathcal{M}^{\pm}} g'$. Suppose $+m \leq_{\mathcal{M}^{\pm}} g$. Then also $+m \leq_{\mathcal{M}^{\pm}} g'$ and $x{\cdot_{\mathcal{M}}}m \leq_{\mathcal{X}} x'{\cdot_{\mathcal{M}}}m$.
- *The case of* (**trans**). Suppose $(x, g) \leq (y, h)$ because $(x, g) \leq (x', g')$ and $(x', g') \leq (y, h)$. Suppose $+m \leq_{\mathcal{M}^{\pm}} g$.
  By inductive hypothesis there exists $+m' \leq_{\mathcal{M}^{\pm}} g'$ such that $x{\cdot_{\mathcal{X}}}m \leq_{\mathcal{X}} x'{\cdot_{\mathcal{X}}}m'$. Also by inductive hypothesis there exists $+n \leq_{\mathcal{M}^{\pm}} h$ such that $x'{\cdot_{\mathcal{X}}}m' \leq_{\mathcal{X}} y{\cdot_{\mathcal{X}}}n$. The result follows by transitivity of $\leq_{\mathcal{X}}$.

The second part is also by induction.

- *The case of* (**absorb**). By (**absorb**) $(x{\cdot_{\mathcal{X}}}m, g) \leq (x, +m \circ_{\mathcal{M}^{\pm}} g)$. Suppose $() \leq_{\mathcal{M}^{\pm}} g$. Then it follows that $+m \leq_{\mathcal{M}^{\pm}} +m \circ_{\mathcal{M}^{\pm}} g$ and it is a fact that $x{\cdot_{\mathcal{M}}}m = x{\cdot_{\mathcal{M}}}m$. If $m \leq_{\mathcal{M}^{\pm}} g$ then we proceed as in the first part of this lemma.
- *The case of* (**compat**). Suppose $(x, g) \leq (x', g')$ because $x \leq_{\mathcal{X}} x'$ and $g \leq_{\mathcal{M}^{\pm}} g'$. Suppose $() \leq_{\mathcal{M}^{\pm}} g$. Then also $() \leq_{\mathcal{M}^{\pm}} g'$ and $x \leq_{\mathcal{X}} x'$. If $m \leq_{\mathcal{M}^{\pm}} g$ then we proceed as in the first part of this lemma.
- *The case of* (**trans**) is similar.

□

6. An example: substitutions

Consider some first-order syntax $\mathcal{L}$ (e.g. $r ::= a \mid \mathsf{f}(r,\dots,r)$ for some term-formers f and variables a) and consider substitution on it $r[a{\mapsto}s]$ [Baader and Nipkow, 1998]. Write Σ for the set of substitutions on $\mathcal{L}$ with functional composition.

Let σ range over elements of Σ. Substitutions form a submonoid of the function space on $\mathcal{L}$, that is, we can view Σ as a subset $\Sigma \subseteq \mathcal{L}^{\mathcal{L}}$ that is closed under functional composition (so if $\sigma, \sigma' \in \Sigma$ then $\sigma \circ \sigma' \in \Sigma$ and maps a to $(\sigma(a))\sigma'$) and contains the identity function id.

Consider Σ as trivially partially ordered, so that $\sigma \leq \sigma'$ if and only if $\sigma = \sigma'$. Similarly consider $\mathcal{L}$ as trivially partially ordered.

Then we can form $\Sigma^{\pm}$ the imaginary group of substitutions, and $\mathcal{L}^{\pm}$ the set of terms lazily acted on by substitutions.

What do these look like?

6.1. *Substitutions just on variables*

Consider the special case where there are no term-formers so that $\mathcal{L}$ is just the set of variables $\{a, b, c, \dots\}$. For brevity write $\bullet$ for $\cdot_{\mathcal{L}^{\pm}}$. Then:

— $a \bullet [a{\mapsto}b] \geq b$.
— $b \bullet [a{\mapsto}b] \geq b$.
— $a \bullet [a{\mapsto}b]^{-1} \geq a$. We deduce this by taking $a \bullet [a{\mapsto}b] \geq b$ and applying $[a{\mapsto}b]^{-1}$ to both sides.
— $a \bullet [a{\mapsto}b]^{-1} \geq b$ is deduced similarly.

In view of the inequalities above, we can think of $a \bullet [a{\mapsto}b]^{-1}$ as being 'like the unordered pair $\{a, b\}$'. However, this is only an analogy; $a \bullet [a{\mapsto}b]^{-1}$ displays behaviour that is *not* like the behaviour of $\{a, b\}$. In particular, $a \bullet [a{\mapsto}b]^{-1} \neq b \bullet [b{\mapsto}a]^{-1}$, yet (by symmetry) both are 'like $\{a, b\}$'.

6.2. *Substitutions on terms*

Now suppose that there are term-formers. Then $r \bullet [a{\mapsto}s] \geq r[s/a]$ (where $r[s/a]$ is the term obtained by really replacing a by s in r). Also, $r \bullet [a{\mapsto}s]^{-1} \geq r'$ for any r' such that $r'[s/a] = r$.

Now consider a rewrite rule $R = (s_1 \to s_2)$ in the sense of first-order rewriting [Baader and Nipkow, 1998]. Then if r rewrites to r' with rule R then $(r \bullet [a{\mapsto}s_1]^{-1}) \bullet [a{\mapsto}s_2] \geq r'$.

For instance, given 0-ary term-formers 1 and 2 (i.e. constant symbols) it is easy to verify that $(1 \bullet [a{\mapsto}1]^{-1}) \bullet [a{\mapsto}2] \geq 2$. This encodes, in a certain sense, the rewrite rule $1 \to 2$.

By combining substitutions and their lazy inverses, it is not hard to encode arbitrary first-order rewriting in $\mathcal{L}^{\pm}$. As an easy corollary, $\leq_{\mathcal{L}^{\pm}}$ is undecidable—it suffices to write out a rewrite system for combinators S and K [Hindley and Seldin, 2008, Section 2].

So equality in Σ is decidable (we just unfold the substitution) but equality in $\Sigma^{\pm}$ is not (given even two term-formers, lazy substitutions can encode undecidable computations).

7. Conclusions

We set off wanting to homomorphically inject every monoid into a group. This is impossible, but we can do something similar which perhaps captures the spirit of this: we can inject every monoid into a partially ordered group.

There is an appealing computational interpretation for this, in which the groups are interpreted as 'executable plans' and the monoids as 'computation steps'.

In order to expand this observation to a categorical adjunction we discover, somewhat surprisingly, an adjunction between the category of partially ordered monoids and lazy homomorphisms and the category of partially ordered groups and homomorphisms (we do not need this latter to be lazy).

We also obtain similar results for sets with an action. This is motivated by the spirit of Cayley's theorem which allows every monoid and group to be represented as the action on a set.[††] These also display the lazy homomorphism/homomorphism structure.

The technical highlights of this paper are in the very specific design of the definitions, notably Definitions 3.8, 3.12, and 5.2, and in the proofs of Theorems 3.22, 3.24 and 5.9. The proofs involve some detailed combinatorial arguments, and which rely on technical lemmas the choice of which is non-evident: specifically these are Lemma 3.14 to Proposition 3.19, and Lemma 5.5 to Lemma 5.8.[‡‡]

In conclusion, let us recall why this is interesting. Monoids are an abstract model of computation, which preserves only the idea that there are some operations (including a 'do nothing' identity operation) and operations can be composed. Groups are a similar model, but for reversible operations.

We have given a sense in which any monoid can systematically be injected into a universe of reversible operations. In that sense, the results in this paper give a new map for moving between irreversible and reversible computation.

Related work

To the best of our knowledge the connection we have observed between partially ordered monoids and lazy morphisms, and partially ordered groups and morphisms, is new. It sheds quite an interesting light on these structures and suggests a new way of (and indeed, a new motivation for) looking at them, one coming from the theory of logic and computation.

Conditions for being able to embed (homomorphically inject) a monoid in a group

[††] Some interesting historical comments on the the attribution of Cayley's name to this family of results is in [Nummela, 1980]—Cayley certainly did not prove it for monoids. Still, 'Cayley's theorem' is the standard word for this kind of result.

[‡‡] Framing the question and then finding the proof structure took two years, from start to finish.

were considered by Mal'cev [Mal'cev, 1937, 1939] and Lambek [Lambek, 1951]; these are infinite sets of Horn clauses that characterise the class of monoids that inject into some group. This was later generalised to categories and groupoids by Johnstone [Johnstone, 2008]. It should be clear that we are solving a distinct problem; we take all monoids, but weaken 'homomorphism/functor' to '*lazy* homomorphism/functor'.[§§]

An inverse semigroup has a natural partial order given by $x \leq y$ when $x = ye$ for some idempotent e. A *prehomomorphism* of inverse semigroups is a function θ such that $\theta(xy) \leq \theta(x)\theta(y)$. Finally, an *inductive groupoid* is a groupoid whose hom-sets form a meet semilattice and an *ordered functor* on them is monotone on the ordering of the hom-sets. This extends by the *Ehresmann-Schein-Nambooripad* theorem to an isomorphism between the categories of inverse semigroups and prehomomorphisms, and inductive groupoids. For full definitions and proofs see [Lawson, 1998]. This theorem expresses an equivalence between two ways of capturing the notion of "partial symmetry ordered by the size of the domain"; one notion expressed by semigroups (untyped) and another by inductive groupoids (typed). There is a similarity to our results, but they are not the same. Superficially, we use monoids (but we could drop the identity) and we map to partially-ordered groups (but we could generalise to categories and locally posetal groupoids). A deeper difference is that in our adjunction the unit is not an identity map; we do not build an isomorphism. That is, our main result is not an equivalence, it is an embedding. It remains to be seen if a variation of our result is possible, which might be more aggressive in quotienting the partially ordered group when mapping back to monoids, and so recover something more like the equivalence theorem above as a special case.

A number of computational systems have been studied which bear reversibility in mind, motivated by three facts:

— Physical systems are time-reversible, so reversible computation is, in fact, the computation of the physical world as we know it. Most recently this is relevant in the design of DNA circuits, where the possibility of a computation running 'backwards' in some chemical soup in the test-tube must be taken into account.
— A corollary of *Landauer's principle* is that reversible computation dissipates less heat (i.e. irreversibility is caused by increasing entropy, which corresponds to erasing information). Thus from the point of view of designing circuits, it is relevant to keep track of where information is lost and heat is generated.
— Quantum computation has aroused much interest and here, the basic model of computation is again reversible (with irreversibility caused by measurement).

Relevant publications are as follows:

— [Toffoli, 1980] introduced the *Toffoli gate*, which is a universal reversible logic gate

[§§] One could study our construction in the special case that the Mal'cev or Lambek conditions are satisfied. If so, then something easier would be sufficient to begin with, since our constructions are orthogonal to whether the monoid is commutative: a *commutative* monoid is embeddable in a group if and only if it satisfies *cancellation*: $xy = xz \Rightarrow y = z$.

(see also [Storme et al., 1999], which studies universal reversible logic gates in general).[¶¶]

— Bennet's classic paper [Bennett, 1973] studied the notion of a reversible Turing machine.

— For reversible computation in biologically inspired computing see [Danos and Krivine, 2004, Phillips and Ulidowski, 2007]; for quantum computation good places to start reading are [Spiller, 1998] and [Kitaev et al., 2002, Section 7].

— The connection between reversibility, entropy, and computation is discussed in [Keyes and Landauer, 1970, Landauer, 1961].

The papers above are about concrete reversible computational structures; our concern is more abstract. We consider how, *given* a possibly irreversible system represented abstractly as a monoid, we can generate a corresponding reversible system represented as a group, which corresponds to it in some natural way.

So on the topic of reversible computation, we hope that the constructions in this paper are interesting theoretical tools but the reader should be clear that we do not pretend to have provided evidence that they are directly computationally useful. In particular, we have not studied the representation of this group aside from that given by its construction. We note that determining the partial order in the group may be strictly more complex than determining equality in the monoid, as Subsection 6.2 illustrates, where we note that $\leq_{\mathcal{M}^{\pm}}$ may be undecidable even if $\leq_{\mathcal{M}}$ is decidable.

Of course, equality of most non-trivial computational systems is undecidable, so this is to be expected and is not the end of the world. We can reasonably hope that the constructions of this paper might be useful as a framework with which to design invertible programming systems; the methodology would be, in principle, to design a non-invertible one and apply $(\text{-})^{\pm}$ (of course there is more to a programming language than that, but the underlying idea stands). We now discuss further future work:

Future work

Briefly, we can suggest the following future work and applications:

— *Generalisation to weaker monoid-like structures:*
We noted in Section 4 that the proofs in this paper seem quite robust, and it should be possible generalise the ideas in this paper to the cases of e.g. commutative monoids, semigroups, and applicative structures. Similarly, a monoid is a special case of a category with one element. It is natural to take the results of this paper and extend them to categories. Thus, given an input category we can generate a category (whose hom-sets are partially ordered) in which every arrow is invertible.

— *Algorithms enriched with inverses:*
Substitutions on syntax form a monoid. Therefore, it is possible to consider syntax enriched with invertible substitutions, as hinted at in Section 6.

[¶¶] So: one can build any circuit out of NAND gates, thus a NAND gate is a universal logic gate; similarly, one can build any *reversible* circuit out of Toffoli gates.

A little more work is needed to enrich syntax with invertible substitutions, but it can be done: we need to build inductive definitions that include inverses of substitutions, but a corollary of Theorem 3.30 is that the 'lazy' functor preserves colimits, thus it preserves least upper bounds of countably ascending chains, thus by general nonsense Theorem 3.30 tells us that we can indeed interleave $(\text{-})^{\pm}$ with the construction of an inductive datatype.

It remains to study the properties of such syntax, e.g. whether unification has most general unifiers. We believe that it does, because substitutions can be inverted.[|||] An algorithm would be theoretically interesting, though not necessarily practical since equality might not be decidable.

Of course, higher-order unification [Dowek, 2001] is also not decidable and that does *not* stop useful heuristics from being constructed that seem to work well in practice, e.g. [Huet, 1975]. So syntax with invertible substitutions is an interesting topic—it might even be useful.

— *Computational models:*
Taking up the comments in the *Related work*, it remains to investigate the application of our construction to monoids or similar structures which are specifically designed to model computation: such as a monoid of functions of the untyped λ-calculus under functional composition; or a monoid of processes under parallel composition [Milner, 1999, Danos and Krivine, 2004]; or the applicative structure of a model of the untyped λ-calculus.

— *Generalisation to richer structures:*
Suppose the monoid has extra structure (e.g. the algebraic study of λ-models has met with some success [Manzonetto and Salibra, 2010]). This structure transfers to the lazy case.

It remains to study how algebraic properties of the monoid carry across; how does the 'algebra' of $\mathcal{M}$ compare with a corresponding '(lazy) algebra' of $\mathcal{M}^{\pm}$? If the partial order on the monoid is part of a domain [Abramsky and Jung, 1994], what are the properties of the partial order on its free group?

— *Lattice properties of the free group:*
On a related note, we can ask whether if $\mathcal{M}$ is a partial order with further structure, such as being a lattice, what further structure $\mathcal{M}^{\pm}$ will have. For instance, elements of a Boolean algebra B form a commutative monoid under conjunction and a lattice under implication ($x \leq y$ when $x \wedge y = x$) so it is interesting to ask what kind of logical entity $B^{\pm}$ would be (possibly the commutative version, as discussed in Section 4). This seems related, at least in spirit, to the *anti-formulae* (formulae which when added to a database *delete* an item from it) considered in [Gabbay et al., 2004]. Is there some sense in which the 'logic' of B extends to a 'logic' of $B^{\pm}$?

[|||] One of the original motivations for this work comes from *nominal unification* [Urban et al., 2004]. This is based on permutations, which are invertible, and because of that nominal unification has most general solutions. The question the first author asked which motivated this line of research was: *Suppose substitutions had inverses too; would something like higher-order unification then have most general solutions?* Preliminary calculations suggest that the answer is *yes*.

Last but not least is this: the proofs in this paper are unambiguous, but a higher-level explanation of why they work is currently lacking. That is, the counting arguments in this paper work, but they do so 'by magic': some general conceptual framework of which the counting arguments are an instance, is an open question.

References

Samson Abramsky and Achim Jung. Domain theory. In *Handbook of Logic in Computer Science: Semantic Structures*, volume 3, pages 1–168. Oxford University Press, 1994.

Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

Charles H. Bennett. Logical reversibility of computation. *IBM journal of Research and Development*, 17(6):525–532, 1973.

Eugenia Cheng and Aaron Lauda. Higher-dimensional categories: an illustrated guide book, August 2004. Notes prepared for IMA workshop.

Vincent Danos and Jean Krivine. Reversible communicating systems. In *Proceedings of the 15th International Conference on Concurrency Theory (CONCUR 2004)*, pages 292–307. Springer, 2004.

Gilles Dowek. Higher-order unification and matching. In *Handbook of automated reasoning*, pages 1009–1062. Elsevier, 2001.

Dov M. Gabbay, Odinaldo Rodrigues, and John Woods. Belief contraction, anti-formulae and resource overdraft: Part II deletion in resource unbounded logics. In *Logic, Epistemology, and the Unity of Science*, volume 1, pages 291–326. Springer, 2004.

Andrew M. W. Glass. *Partially ordered groups*, volume 7 of *Series in Algebra*. World Scientific, 1999.

Maurice Herlihy and J. Eliot B. Moss. *Transactional memory: Architectural support for lock-free data structures*, volume 21(2). ACM Press, 1993.

J. Roger Hindley and Jonathan P. Seldin. *Lambda-Calculus and Combinators, An Introduction*. Cambridge University Press, 2nd edition, 2008.

Gérard Huet. A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1:27–57, 1975.

Peter T. Johnstone. On embedding categories in groupoids. *Mathematical Proceedings of the Cambridge Philosophical Society*, 145:273–294, 2008.

Robert W. Keyes and Rolf Landauer. Minimal energy dissipation in logic. *IBM Journal of Research and Development*, 14:153–157, 1970.

Alexi Yu Kitaev, Alexander H. Shen, and Mikhail N. Vyalyi. *Classical and quantum computation*, volume 47 of *Graduate Studies in Mathematics*. American Mathematical Society, 2002. Translated from the Russian by Lester J. Senechal.

J. Lambek. The immersibility of a semigroup into a group. *Canadian Journal of Mathematics*, 3:3443, 1951.

Rolf Landauer. Irreversibility and heat generation in the computational process. *IBM Journal of Research and Development*, 5:183–191, 1961.

Mark V. Lawson. *Inverse semigroups: the theory of partial symmetries*. World Scientific, 1998.

Tom Leinster. *Higher Operads, Higher Categories*, volume 298 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 2004.

Chuck Liang and Dale Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *Theoretical Computer Science*, 410(46):4747–4768, 2009.

Anatolii I. Mal'cev. On the immersion of an algebraic ring into a field. *Mathematische Annalen*, 113:686–691, 1937.

Anatolii I. Mal'cev. On the embedding of associative systems in groups i. *Matematicheskii Sbornik*, 6(48)(2):331–336, 1939. In German: Über die Einbettung von assoziativen Systemen in Gruppen.

Giulio Manzonetto and Antonino Salibra. Applying universal algebra to lambda calculus. *Journal of Logic and computation*, 20(4):877–915, August 2010. .

Robin Milner. *Communicating and mobile systems: the pi-calculus*. Cambridge University Press, 1999.

Eric Nummela. Cayley's theorem for topological groups. *American Mathematical Monthly*, 87(3):202–203, 1980.

Iain Phillips and Irek Ulidowski. Reversing algebraic process calculi. *Journal of Logic and Algebraic Programming*, 73(1-2):70–96, 2007.

Greg Restall. *An Introduction to Substructural Logics*. Routledge, 1999.

Timothy P. Spiller. Basic elements of quantum information technology. In *Introduction to Quantum Computation and Information*, pages 1–28. World Scientific, 1998.

Leo Storme, Alexis De Vos, and Gerald Jacobs. Group theoretical aspects of reversible logic gates. *Journal of Universal Computer Science (J.UCS)*, 5(5):307–321, 1999.

Tommasso Toffoli. Reversible computing. In *Proceedings of the 7th Colloquium on Automata, Languages and Programming*, pages 632–644. Springer, 1980.

Christian Urban, Andrew M. Pitts, and Murdoch J. Gabbay. Nominal Unification. *Theoretical Computer Science*, 323(1–3):473–497, September 2004. .